1978

The Data Management Subsystem of the System for Performance Evaluation of PDE Software

J. M. Bonnet

R. F. Boisvert

Report Number:
78-286

THE DATA MANAGEMENT SUBSYSTEM

OF THE SYSTEM FOR PERFORMANCE EVALUATION OF PDE SOFTWARE

TR. 286

J. M. Bonnet

R. F. Boisvert

Department of Computer Sciences

Purdue University

August 30, 1978

A substantial amount of data is being collected at Purdue to systematically compare various software modules for the numerical solution of partial differential equations (PDEs) [2]. To ease the performance evaluation effort, a data base system has been implemented to provide efficient access to this data. Part I of this document describes the use of FORTRAN subprograms which provide the information retrieval capability in this data base. Part II describes procedures for updating the data base.

FILE ORGANIZATION

The data whose maintenance is described here has been collected as part of an effort to systematically compare the performance of various software modules for the numerical solution to partial differential equations. To perform these evaluations, we choose a large number of partial differential equations [3], and then use several different numerical methods to solve each of them. For each problem/method combination we make several computer runs, thus generating a table of information which allows us to measure, among other things, the error level achieved as a function of computer resources used. The logical file organization that we have chosen reflects these connections in the data.

The data is stored using a random access file structure based on a locally written random disk file access mechanism [4]. The keys in the main file are record numbers from the EQNFIL file of test problems described in [3]. The actual data is divided into two files, the first (RUNFIL) containing descriptive information of each problem, including a list of which methods have been tested on each problem, and the second (TABFIL) containing the tabular results of all the tests. The logical organization of these files is depicted in Figure 1.

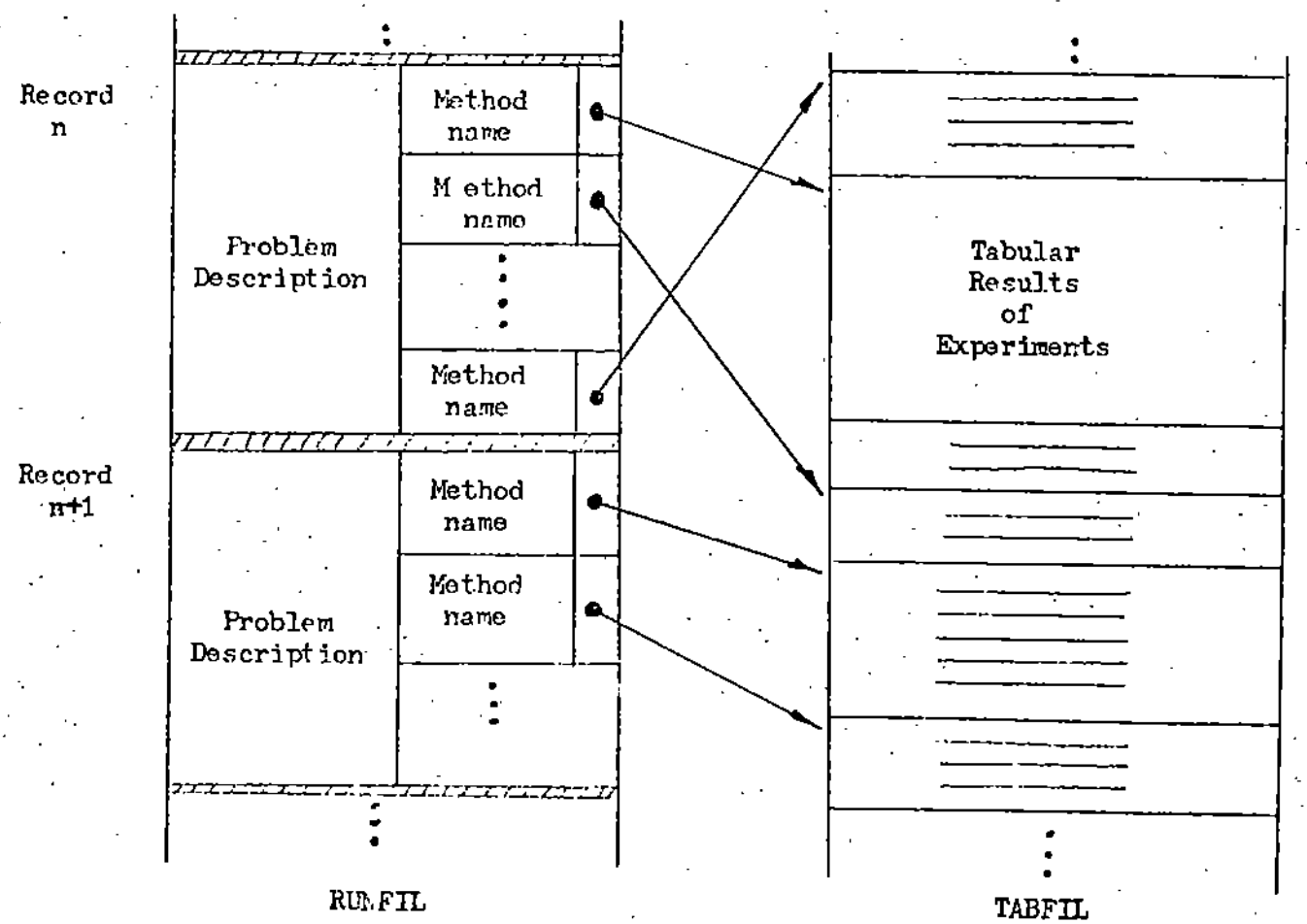

Figure 1 : Logical file organization

PART I : USER INTERFACE

DECK SETUP

To retrieve data from the system, the user must write a FORTRAN program. The following deck setup should be used:

```
MNF(N)
FILES(ELLPK77,T=R)
ATTACH(TABFIL,ELLPK77)
PFILES(GET,RUNFIL,ID=CIC)
PFILES(GET,IRSBIN,ID=CIC)
LOAD(LGO,MNFLIB,RUNLIB)
LOAD(IRSBIN,MNFLIB.RUNLIB)
EXECUTE,,INPUT,OUTPUT,RUNFIL,TABFIL.
7/8/9

    user's FORTRAN program

7/8/9

    input data for user's program if any

6/7/8/9
```

The control cards begin by compling the user's FORTRAN program. Perm-files are then referenced for a copy of TABFIL. The two PFILES calls get copies of RUNFIL and the compiled binary of the data retrieval subprograms (IRSBIN). The user's FORTRAN program and the binary file IRSBIN are then loaded and executed.

Also, the program card for the main program must declare the files OUTPUT, RUNFIL and TABFIL. OUTPUT must be given logical unit number 6 and the other files may be given any other logical unit numbers.

CALLING SEQUENCE

All access to data in this subsystem is performed through the FORTRAN subprogram IRSSM. Its calling sequence is

```
CALL IRSSM(UNIT1,UNIT2,JOB,RECNO,PINFO,PSIZE,METHOD,
           MSIZE,TABUF,TSIZE,NRMETH,ERFLAG)
```

where each variable is of type integer. The following table provides more details.

NAME	INPUT/OUTPUT	NUMBER OF WORDS
UNIT1	INPUT	1
UNIT2	INPUT	1
JOB	INPUT	1
RECNO	INPUT*	1
PINFO	OUTPUT*	11
PSIZE	OUTPUT*	1
METHOD	INPUT/OUTPUT*	8
MSIZE	INPUT/OUTPUT*	1
TABUF	OUTPUT*	4000
TSIZE	OUTPUT*	1
NRMETH	OUTPUT*	1
ERFLAG	OUTPUT	1

* the use of these parameters depends on the values of the JOB selected by the user and hence need not always be specified.

Four of these parameters must be specified on every call. They are

```
UNIT1  =   logical unit number of RUNFIL
UNIT2  =   logical unit number of TABFIL
JOB    =   an integer specifying which job is to be performed
ERFLAG =   an error flag.
           Possible values returned in this variable are

              0 = successful termination
              1 = non-existent record number specified
              2 = non-existent method number specified
              3 = illegal job code
```

Parameters that are not needed for a given job may be replaced by zeros in the CALL statement.

The jobs performed by IRSSM are listed below along with the description of relevant parameters.

JOB=1 : GET TABULAR INFO FOR SPECIFIC RECORD/METHOD

RECNO	input	The desired record number
METHOD	input	The desired method name left justified in 8A10 format with blank fill
MSIZE	input	The size of the array METHOD. Must be at least 8 words
TABUF	output	The tabular information
TSIZE	output	The number of 19-word 'lines' in TABUF

For a description of the tabular information see Appendix I.

JOB=2 : GET PROBLEM INFORMATION OF SPECIFIC RECORD

RECNO	input	The desired record number
PINFO	output	The problem information
PSIZE	output	The number of words of problem information

For a description of the problem information see Appendix I.

JOB=3 : PRINT ALL DATA IN A SPECIFIED RECORD

RECNO	input	The desired record number

JOB=4 : PRINT ALL DATA

No other parameters needed

JOB=5 : GET THE TABULAR INFO FOR THE 'NEXT' METHOD IN SPECIFIED RECORD

RECNO	input	The desired record number
METHOD	output	The name of the method whose tabular information has just been returned, left justified in A10 format with blank fill.
MSIZE	output	Length of the method name in words
TABUF	output	The tabular information
TSIZE	output	The number of 19-word 'lines' in TABUF

This job code allows the user to sequentially process all the tabular information in a given record without knowing the method names in advance. The order in which information is returned is simply the order in which it has been stored.

ERFLAG=2 signals that all the methods have been processed. Note that if any other request is made between two calls with JOB=5, then the 'next' method to be processed will be the first.

JOB=6 : COUNT NUMBER OF METHODS IN SPECIFIED RECORD

RECNO	input	The desired record number
NRMETH	output	The number of distinct methods for whom data has been saved in this record.

Errors detected by IRSSM cause a message to be printed on the output file. The request causing the error is ignored, and control returns to the call routine for the next request. The error flag is also set for user detection. Calls to IRSSM need contain only those parameters of interest since IRSSM uses only the parameters indicated by the job code. Requests indicating the need for buffers to return information where none are provided give undefined results.

EXAMPLES OF INFORMATION RETRIEVAL

Example 1

```
      PROGRAM TEST(INPUT,OUTPUT,RUNFIL,TABFIL,TAPE5=INPUT,
     $            TAPE6=OUTPUT,TAPE7=RUNFIL,TAPE8=TABFIL)

      INTEGER ERFLAG

        TO OBTAIN A COMPLETE DUMP OF THE ENTIRE FILE WE SIMPLY SET
        JOBTODO TO FOUR AND IRSSM WILL PRINT THE WHOLE FILE TO OUTPUT
        ON TAPE6.

      CALL IRSSM(7,8,4,0,0,0,0,0,0,0,0,ERFLAG)

      STOP
      END
```

Example 2

```
      PROGRAM TEST(INPUT,OUTPUT,RUNFIL,TABFIL,TAPE5=INPUT,
     $            TAPE6=OUTPUT,TAPE7=RUNFIL,TAPE8=TABFIL)

      INTEGER NRMETH,ERFLAG

C
C       TO RECIEVE A COUNT OF THE NUMBER OF METHODS FOR A GIVEN
C       RECORD NUMBER SET RECORD NUMBER AND SET JOBTODO.  THE COUNT
C       WILL BE RETURNED IN NRMETH.
C
      CALL IRSSM(7,8,6,22,0,0,0,0,0,0,NRMETH,ERFLAG)

C
C       GIVEN COUNT RETURNED FOR RECORD
C
      WRITE(6,1) NRMETH
      STOP
    1 FORMAT(≠0≠,≠THE NUMBER OF METHODS FOR THE RECORD IS ≠,I4)
      END
```

```
 THE NUMBER OF METHODS FOR THE RECORD IS    7
```

Example 3

```
      PROGRAM TEST(INPUT,OUTPUT,RUNFIL,TABFIL,TAPE5=INPUT,
     $            TAPE6=OUTPUT,TAPE7=RUNFIL,TAPE8=TABFIL)

      INTEGER PINFO(41),PBUFLEN


C
C       TO GET ONLY THE PROBLEM INFORMATION FOR THE RECORD RETURNED
C       WE SET THE RECORD NUMBER AND PASS STORAGE VARIABLES FOR THE
C       INFORMATION TO BE RETURNED IN.
C
      CALL IRSSM(7,8,2,22,PINFO,PBUFLEN,0,0,0,0,0,ERFLAG)

C
C       SHOW THAT RESULTS WHERE RETURNED IN THE BUFFER
C
      WRITE(6,1) PINFO(1),PINFO(2),(PINFO(I),I=9,11),
     $           (PINFO(I),I=3,8)

      STOP
    1 FORMAT(≠0≠,≠RECORD = ≠,I5,/≠ ≠,≠PROBLEM = ≠,I5,3X,≠(≠,3A10,≠)≠,
     $       /≠ ≠,≠PROBLEM INFO = ≠,3X,2A10,3X,4A10)
      END
```

```
RECORD =    22
PROBLEM =     9   (A=100.0,B=0.5                    )
PROBLEM INFO =    202022100200022        009.10    000.10    000.00    030.40
```

Example 4

```
      PROGRAM TEST(INPUT,OUTPUT,RUNFIL,TABFIL,TAPE5=INPUT,
     $            TAPE6=OUTPUT,TAPE7=RUNFIL,TAPE8=TABFIL)

      INTEGER ERFLAG


C
C       TO HAVE ALL THE INFORMATION FOR A WHOLE PROBLEM PRINTED
C       WE SIMPLY PASS THE RECORD NUMBER AND THE UNIT NUMBERS OF THE
C       RUNFIL AND TABFIL AND THE ENTIRE RECORD IS PRINTED FOR US.
C
      CALL IRSSM(7,8,3,22,0,0,0,0,0,0,0,ERFLAG)

      STOP
      END
```

The output from this run appears on the next page.

RECORD = 22
PROBLEM = 9 (A=100.0,B=0.5)
PROBLEM INFO = 202022100200022 009.10 000.10 000.00 030.40

METHOD = 5,IORDER=4/9/16/

DATE	TIME	NX	NY	HMAX	NRUNK	ERRMAX	ERRMXF	ERRL2	RESMAX	RESMXR	RESL2	SOLMAX	NIT	MEM	TOTLT	TIME1	TIME2	TIME3
04/18/78	14.12	3	3	5.00E-01	1	9.0E+00	0	1.0E+00	1.6E+02	9.3E+03	8.0E+01	1.0E+01	0	6762	.10	.10	0	0
04/18/78	14.12	5	5	2.50E-01	9	1.3E+00	0	5.5E-02	1.7E+02	2.9E+08	1.1E+01	2.3E+00	0	7042	.16	.14	.01	.01
04/18/78	14.12	7	7	1.67E-01	25	3.2E-01	0	1.7E-02	1.7E+02	8.9E+09	5.0E+00	6.8E-01	0	7690	.30	.20	.05	.04
04/18/78	14.12	9	9	1.25E-01	49	1.3E-01	0	2.6E-03	1.5E+02	1.2E+10	2.8E+00	8.7E-01	0	8802	.52	.28	.10	.14
04/20/78	17.42	11	11	1.00E-01	81	3.8E-02	0	3.8E-04	1.4E+02	6.5E+09	2.7E+00	9.6E-01	0	10474	.90	.39	.18	.33

METHOD = 2/10/16/

DATE	TIME	NX	NY	HMAX	NRUNK	ERRMAX	ERRMXF	ERRL2	RESMAX	RESMXR	RESL2	SOLMAX	NIT	MEM	TOTLT	TIME1	TIME2	TIME3
04/18/78	14.29	3	3	5.00E-01	36	3.6E+00	0	4.0E-01	1.5E+01	8.9E+02	6.9E+00	2.6E+00	0	5554	.43	.08	.03	.32
04/18/78	14.29	5	5	2.50E-01	100	5.7E-01	0	4.8E-02	5.4E+01	2.3E+08	6.6E+00	1.6E+00	0	12216	2.08	.23	.10	1.75
04/18/78	14.29	7	7	1.67E-01	196	4.4E-01	0	1.6E-02	6.4E+01	3.2E+10	3.5E+00	1.4E+00	0	24478	6.30	.44	.21	5.65
04/20/78	17.43	8	8	1.43E-01	256	1.5E-01	0	5.8E-03	2.1E+02	8.1E+10	4.3E+00	2.1E-01	0	33189	9.93	.59	.29	9.05

Example 5

```
      PROGRAM TEST(INPUT,OUTPUT,RUNFIL,TABFIL,TAPE5=INPUT,
     $             TAPE6=OUTPUT,TAPE7=RUNFIL,TAPE8=TABFIL)

      INTEGER METHOD(8),TABUF(4000),INFOLEN,ERFLAG,IENTRY(19,50),
     $        BLANK

      REAL    RENTRY(19,50)

      EQUIVALENCE ( TABUF,IENTRY,RENTRY)

      DATA BLANK/10H          /

C
C       SET THE METHOD TO THE METHOD DESIRED FOR THE PROBLEM
C       TO BE CALLED.  THE METHOD STRING SHOULD BE CHARACTER
C       SET UP IN TEN CHARACTER/WORD STRINGS.
C
      DO 10 I= 1,8
         METHOD(I) = BLANK
   10 CONTINUE

      METHOD(1) = ≠5,IORDER=4≠
      METHOD(2) = ≠/9/16/≠

C
C       NEXT CALL IRSSM GIVING FILE UNIT NUBERS AND RECORD NUMBER.
C       ALSO PASS THE METHOD, AND GIVEN PARAMETERS FOR RETURNING
C       THE TABLE INFORMATION.
C
      CALL IRSSM(7,8,1,12,0,0,METHOD,0,TABUF,INFOLEN,0,ERFLAG)

C
C       WRITE OUT RESULTS OF REQUEST
C
      WRITE(6,1) (METHOD(I),I=1,8)
      WRITE(6,2) ((IENTRY(I,J),I=1,4),RENTRY(5,J),IENTRY(6,J),
     $             (RENTRY(I,J),I=7,9 ),
     $             (RENTRY(I,J),I=16,19),J=1,INFOLEN)

      STOP
    1 FORMAT(////7X,9HMETHOD = ,8A10//1X,
     $       127(1H-)/3X,4HDATE,3X,4HTIME,3X,2HNX,2X,2HNY,3X,4HHMAX,
     $       3X,5HNRUNK,2X,6HERRMAX,2X,6HERRMXF,2X,5HERRL2,3X,
     $       5HTOTLT,1X,5HTIME1,1X,5HTIME2,1X,5HTIME3 /1X,127(1H-) / )
    2 FORMAT(1X,A8,1X,A5,1X,2(I3,1X),E8.2,1X,I5,1X,3(E7.1,1X),
     $       4(1X,F5.2) )
      END
```

METHOD = 5,IORDER=4/9/16/

DATE	TIME	NX	NY	HMAX	NRUNK	ERRMAX	ERRMXF	ERRL2	TOTLT	TIME1	TIME2	TIME3
04/18/78	10.43	3	3	5.00E-01	1	1.8E-02	0	2.0E-03	.10	.10	0	0
04/18/78	10.43	5	5	2.50E-01	9	3.4E-03	0	2.0E-04	.17	.15	.01	.01
04/18/78	10.43	7	7	1.67E-01	25	7.8E-04	0	3.9E-05	.30	.21	.05	.04
04/18/78	10.43	9	9	1.25E-01	49	2.6E-04	0	1.1E-05	.54	.30	.11	.13
04/19/78	15.29	11	11	1.00E-01	81	1.1E-04	0	3.8E-06	.93	.41	.18	.34

PART II : FILE UPDATING

INTRODUCTION

In this section we describe the use of a FORTRAN program which allows the addition, deletion and changing of information in the data base. One consequence of the organization chosen for the data base is that when these updates are performed, some space within the files RUNFIL and TABFIL is lost. To recover this lost space a fourth facility, garbage collection, is also available to compact these files.

Caution Although a substantial amount of input checking has been implemented in the update system, it is not foolproof. Incorrect specification of update directives may destroy the data base.

To update the data base the user must prepare a command input file (CFILE) which contains directives specifying what data is to be added, changed or deleted in the data base. When this file has been created and checked, the following control card procedure should be run to create a new data base with the specified changes (we assume that CFILE has been made a local file).

```
XEQ(UPDATE,ID=CIB,I=CFILE)
```

This control card procedure obtains the compiled binary of the creation and maintenance program and the files containing the existing data base (RUNFIL and TABFIL). Upon execution, the command input file and a description of all the changes applied to the data base are listed. Upon successful execution, the local files RUNFIL and TABFIL contain the modified data base. These files are not saved by the XEQ procedure. The user should verify that a correct update has occurred before replacing the permanent copies of RUNFIL and TABFIL. If GARBAGE collection is performed during the run, it is the users responsibilty to correctly handle the newly created data base files RUNNEW and TABNEW (see the description of the GARBAGE directive). If the command file is on INPUT, then the I parameter may be omitted.

UPDATE DIRECTIVES

The directives to the update program are signalled by a card with an asterisk in column one. The command name appears on the next card followed by several optional parameters which depend upon the command. A description of each command follows.

The INSERT Directive

The INSERT directive permits the addition of information to the data base.

```
Directive syntax : 12345678901234567890123456789O
                   *
                   INSERT
                   < data >
```

Data for an arbitrary number of problem/method combinations may be included after a single INSERT directive. The format of < data > is the same as the performance data generated by the ELLPACK system. For details see [1], [2] and Appendix II.

If data is to be inserted for a record number which does not exist in the data base, then a new record is created in RUNFIL. If the data to be inserted is for a method which does not yet exist in the specified record, then the new method is added to the record in RUNFIL. If the record and method already exist, then no new information is added to RUNFIL. In each case, the supplied tabular information is added to TABFIL.

One result of the operation of this directive is that INSERT may be used to create a new data base if the files RUNFIL and TABFIL are empty.

The DELETE Directive

The DELETE directive allows the deletion of all the data associated with a single method, including its associated tabular information, from the data base. If the method is found to exist, then the pointer in the file RUNFIL to the tabular information in TABFIL is destroyed and the information is lost.

```
Directive syntax :  12345678901234567890123456789O
                    *
                    DELETE   RRR
                    < method name >
                    < blank card >
```

RRR is the record number in which the method is to be deleted. The < method name > represents the three-part encoded method name as described in [1] and Appendix II. A blank card must follow the method name.

The CHANGE Directive

The CHANGE directive allows one to replace or delete specific lines of tabular information stored in the data base.

```
Directive syntax :  12345678901234567890123456789O
                    *
                    CHANGE    RRR    SSS    NNN
                    < method name >
                    MM/DD/YY  HH:MM  (if required)
                    < tabular information >   (if required)
                    < blank card >
```

RRR is the record number in which tabular information is to be replaced. < method name > is the three-part name of the method whose tabular information is to be replaced. SSS is the number of the first line of the tabular information to be affected by the change. The presence of the other parameters depend on the value of NNN specified on the CHANGE card.

Case 1 : NNN>0

This specifies that NNN consecutive lines of tabular information is to be replaced with the NNN lines of tabular information which follow (see Appendix II for a description of the format of this information). In addition, the date and time of the runs which this new data represents (MM/DD/YY and HH:MM) must be included.

Case 2 : NNN<0

This specifies that -NNN consecutive lines of tabular information are to be deleted from the data base. The other parameters are not supplied in this case.

The GARBAGE Directive

The GARBAGE directive causes the current data base to be copied from RUNFIL and TABFIL to two new files, RUNNEW and TABNEW. However, the wasted space in RUNFIL and TABFIL which is caused by additions and deletions to the files is not copied. This results in a compacted copy of the data base. This directive, when present, should be the last directive in the command input file, since update directives following it are not applied to the file RUNNEW and TABNEW.

```
Directive syntax :  12345678901234567890123456789O
                    *
                    GARBAGE
                    < blank line >
```

It is the responsibility of the user to verify that the new data base is correct and to replace the old data base with the new one.

EXAMPLE OF FILE UPDATE

The figures on the next three pages give a simple example of the use of the update system. Figure 2 shows the data base before the update. Figure 3 displays the command input file of update directives. Finally, Figure 4 shows the data base after the update has been performed.

```
RECORD =     3
PROBLEM =    3   (                                   )
PROBLEM INFO =    202022100200222          000.02     004.00     000.00     006.00


      METHOD = 5,IORDER=41/9/16/

 DATE    TIME    NX  NY    HMAX     NRUNK  ERRMAX   ERRMXF   ERRL2    RESMAX   RESMXR   RESL2    SOLMAX NIT  MEM   TOTLT
08/04/78 11.30    6   6 2.00E-01     16 4.4E-04 3.7E-02 3.2E-05 3.4E+00 3.7E+00 3.7E-01 5.7E-01   0  7314   .21
08/04/78 11.30    8   8 1.43E-01     36 1.1E-04 1.7E-02 6.5E-06 4.3E+00 3.5E+00 2.0E-01 5.6E-01   0  8182   .38


      METHOD = 2/10/16/

 DATE    TIME    NX  NY    HMAX     NRUNK  ERRMAX   ERRMXF   ERRL2    RESMAX   RESMXR   RESL2    SOLMAX NIT  MEM   TOTLT
08/04/78 11.35    4   4 3.33E-01     36 4.5E-04 1.6E-03 3.9E-05 3.2E-01 8.2E-02 6.9E-02 5.6E-01   0  8281   .31
08/04/78 11.35    5   5 2.50E-01     64 1.4E-04 5.3E-04 1.1E-05 1.8E-01 6.0E-02 2.8E-02 5.1E-01   0 12216   .66
08/04/78 11.35    6   6 2.00E-01    100 5.5E-05 2.3E-04 4.0E-06 1.2E-01 4.7E-02 1.4E-02 5.7E-01   0 17551  1.25


      METHOD = 4,IORDER=4///

 DATE    TIME    NX  NY    HMAX     NRUNK  ERRMAX   ERRMXF   ERRL2    RESMAX   RESMXR   RESL2    SOLMAX NIT  MEM   TOTLT
04/17/78 15.35    9   9 1.25E-01     49 6.8E-05        0 3.5E-06 4.4E+00 3.5E+00 2.0E-01 5.7E-01   0  3504   .05
04/17/78 15.35   17  17 6.25E-02    225 4.3E-06        0 1.2E-07 3.6E+00 3.2E+00 5.2E-02 5.7E-01   0  5296   .18
04/17/78 15.35   33  33 3.13E-02    961 2.7E-07        0 4.0E-09 2.3E+00 3.1E+00 1.3E-02 5.7E-01   0 11952   .71


RECORD =     55
PROBLEM =    41   (A=10.0,B=10.0                      )
PROBLEM INFO =    200022100200222          000.11     006.30     000.00     006.30


      METHOD = 5,IORDER=4/9/16/

 DATE    TIME    NX  NY    HMAX     NRUNK  ERRMAX   ERRMXF   ERRL2    RESMAX   RESMXR   RESL2    SOLMAX NIT  MEM   TOTLT
08/04/78 11.58    4   4 3.33E-01      4 2.0E-01 2.3E+00 1.8E-02 1.8E+02 4.3E+00 2.7E+01 1.4E+00   0  6862   .14
08/04/78 11.58    6   6 2.00E-01     16 3.3E-02 1.4E+00 2.1E-03 3.5E+02 1.9E+02 3.5E+01 2.0E+00   0  7314   .24
```

Figure 2 : The data base before application of the updates.

```
*
INSERT

  3  08/04/78 11.30.44.  $TEST-4TH-ORDER-HODIE
   3(                          )  202022100200222     000.02 004.00 000.00 006.00
5,IORDER=41/9/16/
    4   4   3.33E-01     4   3.51E-03 9.05E-02 3.19E-04 5.93E+00 4.14E+00 1.13E+00 5.66E-01   0  6862    .12   .11   .01   .00
    5   5   2.50E-01     9   1.02E-03 5.68E-02 8.85E-05 4.38E+00 3.88E+00 6.88E-01 5.11E-01   0  7042    .16   .13   .01   .01


*
CHANGE    055       001       001
5,IORDER=4/9/16/
08/04/78  11.58
    8   8   1.43E-01    36   1.14E-04 1.73E-02 6.53E-06 4.30E+00 3.51E+00 1.95E-01 5.64E-01   0  8182    .38   .23   .07   .08

*
CHANGE    003       002       -02
2/10/16/

*
DELETE    003
4,IORDER=4///

*
GARBAGE
```

Figure 3 : The update directives

```
RECORD =     3
PROBLEM =    3   (                                   )
PROBLEM INFO =    202022100200222        000.02    004.00    000.00    006.00


      METHOD = 5,IORDER=41/9/16/

-------------------------------------------------------------------------------------------------------------------------------
  DATE    TIME    NX  NY    HMAX     NRUNK  ERRMAX   ERRMXF   ERRL2    RESMAX   RESMXR   RESL2    SOLMAX  NIT  MEM   TOTLT
-------------------------------------------------------------------------------------------------------------------------------

08/04/78 11.30    4   4 3.33E-01       4 3.5E-03 9.1E-02 3.2E-04 5.9E+00 4.1E+00 1.1E+00 5.7E-01    0  6882   .12
08/04/78 11.30    5   5 2.50E-01       9 1.0E-03 5.7E-02 8.9E-05 4.4E+00 3.9E+00 6.9E-01 5.1E-01    0  7042   .16
08/04/78 11.30    6   6 2.00E-01      16 4.4E-04 3.7E-02 3.2E-05 3.4E+00 3.7E+00 3.7E-01 5.7E-01    0  7314   .21
08/04/78 11.30    8   8 1.43E-01      36 1.1E-04 1.7E-02 6.5E-08 4.3E+00 3.5E+00 2.0E-01 5.6E-01    0  8182   .38


      METHOD = 2/10/16/

-------------------------------------------------------------------------------------------------------------------------------
  DATE    TIME    NX  NY    HMAX     NRUNK  ERRMAX   ERRMXF   ERRL2    RESMAX   RESMXR   RESL2    SOLMAX  NIT  MEM   TOTLT
-------------------------------------------------------------------------------------------------------------------------------

08/04/78 11.35    4   4 3.33E-01      36 4.5E-04 1.6E-03 3.9E-05 3.2E-01 8.2E-02 6.9E-02 5.6E-01    0  8281   .31



RECORD =    55
PROBLEM =    41   (A=10.0,B=10.0                     )
PROBLEM INFO =    200022100200222        000.11    006.30    000.00    006.30


      METHOD = 5,IORDER=4/9/16/

-------------------------------------------------------------------------------------------------------------------------------
  DATE    TIME    NX  NY    HMAX     NRUNK  ERRMAX   ERRMXF   ERRL2    RESMAX   RESMXR   RESL2    SOLMAX  NIT  MEM   TOTLT
-------------------------------------------------------------------------------------------------------------------------------

08/04/78 11.58    6   6 2.00E-01      16 3.3E-02 1.4E+00 2.1E-03 3.5E+02 1.9E+02 3.5E+01 2.0E+00    0  7314   .24
08/04/78 11.58    8   8 1.43E-01      36 1.1E-04 1.7E-02 6.5E-06 4.3E+00 3.5E+00 2.0E-01 5.6E-01    0  8182   .38
```

Figure 4 : The data base after the update

APPENDIX I

The following describes the problem and table information which is returned by IRSSM in the buffers PINFO and TABUF.

The Problem Information

PINFO WORD NUMBER	NAME	TYPE	DESCRIPTION
1	RECNO	Integer	The record number.
2	PROBNO	Integer	The problem number.
3-4	CINFO	Character	Problem/method compatibility info. Fifteen digits of character information in A10,A5 format. See [1].
5-8	PTYPE	Character	Problem type info. Four words in 4A5 format. Each word is in the form xx.xxx, where x is a digit. See [3].
9-11	COMMENT	Character	The parameters of this problem in free format (3A10).

The Tabular Information

TABULAR WORD NUMBER	NAME	TYPE	DESCRIPTION
1	RDATE	Character	Date of run in A8 format.
2	RTIME	Character	Time of run in A5 format.
3	NGRIDX	Integer	Number of grid lines in x direction.
4	NGRIDY	Integer	Number of grid lines in y direction.
5	HMAX	Real	Max grid spacing in any direction
6	NUMBEQ	Integer	Number of unknowns.
7	ERRMAX	Real	Max error at nodes.
8	ERRMXF	Real	Max error on a fixed 20 x 20 grid.
9	ERRL2	Real	Discrete L2 error at nodes.
10	RESMAX	Real	Max residual at midpoints of subrectangles.
11	RESMXR	Real	Max relative residual at midpoints.
12	RESL2	Real	Discrete L2 residual at midpoints.
13	SOLMAX	Real	Max values of computed solution at nodes.
14	NRITNS	Integer	Number of iterations for iterative method.
15	MEMORY	Integer	Number of memory words used.
16	TLTIME	Real	Total execution time.
17-19	TIME	Real	Individual module times.

These 19 words of tabular information are repeated TSIZE times, once for each line of the table that has been returned. The table is sorted in decreasing order of HMAX.

In order to ease the use of TABUF it should be declared as

```
INTEGER TABUF(19,50)
```

in the main program. In this way, for example,each value of NUMBEQ, which is the 6th word in each line, may be obtained by referencing the locations TABUF(6,K), K=1,....,TSIZE.

Finally, we note that both real and integer information has been returned in the integer array TABUF. Thus, to avoid type conversions, it is nessary to equivalence a real array to TABUF whose name should be used when referencing the elements in TABUF of type real. This can be done as follows

```
REAL RTABUF(19,50)

EQUIVALENCE(TABUF,RTABUF)
```

So, for example, to obtain all values of RESL2 that have been returned, one should reference RTABUF(12,K),K=1,...TSIZE.

APPENDIX II

The following describes the format of the input to the INSERT directive. This data is in the same form as the performance data generated by the ELLPACK system [2]. Note that if multiple sets of data are used with a single INSERT, then the end of each set of tabular information must be signalled by a blank card. The tabular information in the data is in the format expected by the CHANGE directive. We use the same variable names as Appendix I.

Problem/method Information

```
RECNO,RDATE,RTIME,
PROBNO,(COMMENT(I),I=1,3),(CINFO(I),I=1,2),(PTYPE(I),I=1,2),
(METHOD(I),I=1,8)

FORMAT :  /I4,2X,A8,1X,A9
          /I4,1H(,3A10,1H),3X,A10,A5,4X,4(1X,A5)
          /8A10
```

Tabular Information

An arbitrary number of lines of the form :

```
NGRIDX,NGRIDY,HMAX,NUMBEQ,ERRMAX,ERRMXR,ERRL2,RESMAX,RESMXR,
RESL2,SOLMAX,NRITNS,MEMORY,TLTIME,(TIME(I),I=1,3)

Format : 1X,2I4,1X,E10.2,1X,I5,2X,7E9.2,1X,I3,1X,I5,1X,4F6.2
```

Method Names

The method name is an encoded description of the sequence of modules executed in an ELLPACK run. It is of the form D/I/S/, where D represents the discretization module, I represents the indexing module and S represents the solution module. Allowable values of D, I and S are specified in [1].

REFERENCES

1. Boisvert, R. F., A description of the testing procedure for the evaluation of methods for solving PDEs, July 26, 1978.

2. Boisvert, R. F., Houstis, E. N. and Rice, J. R., A system for the performance evaluation of partial differential equations software, Purdue University Computer Sciences Department Report CSD-TR 278, July 1978.

3. Houstis, E. N. and Rice, J. R., A population of partial differential equations for evaluating methods, Purdue University Computer Sciences Department Report CSD-TR 263, May 15, 1978.

4. Random I/O, Purdue University Computer Center Document K3 RANDIO, Aug. 1975.